Bidirectionally-Secure Cryptosystems

YEVGENIY DODIS[*] JONATHAN KATZ[†] SHOUHUAI XU[‡] MOTI YUNG[§]

January 11, 2002

Abstract

Protecting private keys is perhaps the most basic requirement of any cryptographic scheme. On the other hand, cryptographic computations (decryption, signature generation, etc.) are often performed on a relatively insecure device (e.g., a mobile device or an Internet-connected host) which cannot be trusted to maintain secrecy of the private key. We propose and investigate the notion of *bidirectional security* whose goal is to minimize the damage caused by secret-key exposure. In our model, the secret key(s) stored on the insecure device are refreshed at discrete time periods via interaction with a physically-secure — but computationally-limited — device which stores a "master key". All cryptographic computations are still done on the insecure device, and the public key remains unchanged. In a (t, N)-bidirectionally-secure scheme, an adversary who compromises the insecure device and obtains secret keys for up to t periods of his choice is unable to violate the security of the cryptosystem for *any* of the remaining $N - t$ periods. Furthermore, the scheme remains secure (for *all* time periods) against an adversary who compromises *only* the physically-secure device.

Bidirectionally-secure schemes significantly improve the security guarantee of forward-secure schemes [3, 5], in which exposure of the secret key at even a single time period (necessarily) compromises the security of the system for all future time periods. This improvement is achieved with minimal cost: infrequent key updates with a (possibly untrusted) secure device.

We focus primarily on bidirectionally-secure public-key encryption. We construct a (t, N)-bidirectionally-secure encryption scheme based on any (standard) public-key encryption scheme, and also give a more efficient construction based on the DDH assumption. Our second construction can be improved to achieve chosen-ciphertext security as well.

[*]dodis@cs.nyu.edu. Department of Computer Science, NYU.

[†]jkatz@cs.columbia.edu. Department of Computer Science, Columbia University. This work was done while the author was at Telcordia Technologies.

[‡]shouhuai@yahoo.com. Department of Computer Science, George Mason University.

[§]moti@cs.columbia.edu. CertCo, Inc.

1 Introduction

Exposure of secret keys is perhaps the most devastating attack on a cryptosystem since it typically means that security is entirely lost. This problem is probably the greatest threat to cryptography in the real world: in practice, it will be much easier for an adversary to obtain a secret key from a naive user than to break the computational assumption on which the system is based. The threat is increasing nowadays with users carrying mobile devices which allow remote connections from public or foreign domains.

Two classes of methods exist to deal with this problem. The first tries to prevent key exposure altogether. While this is an important goal, it is not always practical. For example, when using portable devices to perform cryptographic operations (e.g., decrypting transmissions using a mobile phone) one must expect that the device itself may be physically compromised in some way (e.g., lost or stolen) and thus key exposure is inevitable. Furthermore, complete prevention of key exposure — even for non-mobile devices — will usually require some degree of physical security which can be expensive and inconvenient. The second approach assumes that key exposure will inevitably occur and seeks instead to minimize the damage which results when keys are obtained by an adversary. Much recent work has focused on providing elegant and efficient solutions in this case.

The most successful solution involves a combination of the above approaches. Physical security may be ensured for a *single* device and thus we may assume that data stored on this device will remain secret. On the other hand, this device may be computationally limited or else not suitable for a particular application and thus we are again faced with the problem that some keys will need to be stored on insecure devices which are likely to be compromised during the lifetime of the system. Therefore, techniques to minimize the damage caused by such compromises must also be implemented.

We focus here on a notion we term *bidirectional security*. Our model is the following (the discussion here focuses on public-key encryption, yet the term applies equally-well to the case of digital signatures; a formal definition appears in Section 2): the user begins by registering a single public key PK. A "master" secret key SK^* is stored on a device which is physically secure and hence resistant to compromise. All decryption, however, is done on an insecure device for which key exposure is expected to be a problem. The lifetime of the protocol is divided into distinct periods $1, \ldots, N$ (for simplicity, one may think of these time periods as being of equal length; e.g., one day). At the beginning of each period, the user interacts with the secure device to derive a temporary secret key which will be used to decrypt messages sent during that period; we denote by SK_i the temporary key for period i. On the other hand, the public key PK used to encrypt messages does *not* change at each period; instead, ciphertexts are now labeled with the time period during which they were encrypted. Thus, encrypting M in period i results in ciphertext $\langle i, C\rangle$.

The insecure device, which does all actual decryption, is vulnerable to repeated key exposures. Our goal is to minimize the effect such compromises will have. Of course, when a key SK_i is exposed, an adversary will be able to decrypt messages sent during time period i. Our notion of security (informally) is that this is all an adversary can do. In particular, the adversary will be unable to determine any information about messages sent during time periods other than those in which a compromise occurred. This is the strongest level of security one can expect in such a model.

If the physically-secure device is completely trusted, this device may generate (PK, SK^*) itself, keep SK^*, and publish PK. When a user requests a key for period i, the device computes SK_i and sends it to the user. More involved methods are needed when the physically-secure device is *not* trusted by the user. In this, more difficult case (which we consider here), the user may generate (PK, SK) himself, publish PK, and then derive keys SK^*, SK_0. The user then sends SK^* to the device and stores SK_0 himself. When the user requests a key for period i, the device sends "partial" key SK'_i to the user, who may then compute the "actual" key SK_i using SK_{i-1} and SK'_i. In this way, the user's security is guaranteed, during all time periods, with respect to the device itself. This security guarantee is essential when a single device serves many different users, offering them protection against key exposure. In this scenario, users may trust this device to update their keys, but may not want the device to be able to read their encrypted traffic. Thus, there is no reason this device should have complete (or *any*!) knowledge of their "actual" keys.

Other Applications. Besides the obvious application to minimizing the risk of key exposures across multiple time periods, bidirectional security may also be used to protect against key exposures across multiple locations, or users. For example, a company may establish a single public key and distribute (different) secret keys to its various employees; each employee is differentiated by its "non-cryptographic ID" i (e.g., a simple counter), and can use his own secret key SK_i to perform the desired cryptographic operation. This approach could dramatically save on the public key storage, and has the property that the system remains secure (for example, encrypted messages remain hidden) for all employees whose keys are not exposed.

A bidirectional scheme may also be used for purposes of delegation [18]; here, a user (who has previously established a public key) delegates his rights in some specified, limited way to a second party. In this way, even if a certain number of the delegated parties' keys are lost, the remaining keys — and, in particular, the user's secret key —- are secure.

Finally, we mention the application of key escrow by legal authorities. For example, consider the situation in which the FBI wants to read email sent to a particular user on a certain date. If a bidirectional scheme (updated daily) is used, the appropriate key for up to t desired days can be given to the FBI without fear that this will enable the FBI to read email sent on other days. A similar application (with weaker security guarantees) was considered by [2].

Our Contributions. We introduce the notion of bidirectional security and construct efficient schemes secure under this notion. Although our definition may be applied to a variety of cryptographic primitives, we focus here on public-key encryption and digital signatures. In Section 3.1, we give a generic construction of a (t, N)-bidirectionally-secure encryption scheme based on any (standard) public-key encryption scheme. Section 3.2 gives a more efficient construction which is secure under the DDH assumption. Both of these scheme achieve semantic security; however, we show in Section 3.3 how the second scheme can be improved to achieve chosen-ciphertext security. In a companion paper, we consider bidirectional security of signature schemes [32].

1.1 Previous Work

Arriving at the right definitions and models for the notion we put forth here has been somewhat elusive. In the context of developing smart-card applications, the preliminary work of Girault [17] considers a notion similar to bidirectional security of signatures, and presents schemes designed to minimize effects of key loss during a limited number of periods. However, [17] does not present any formal definitions, nor does it present schemes which are provably secure.

Recently and concurrently with our work (motivated by the notion of forward security, see below) other attempts at formalizing bidirectionally-secure public-key encryption have taken place [31, 26]. However, various problems are present in these works. First, these works consider only a *non-adaptive* adversary who chooses which time periods to expose at the outset of the protocol, whereas we consider the more natural and realistic case of an *adaptive* adversary who may choose which time periods to expose at any point during protocol execution. Furthermore, the solution of [31] for achieving chosen-ciphertext security is proven secure in the random oracle model; our construction of Section 3.3 is proven secure against chosen-ciphertext attacks in the standard model ([26] does not address chosen-ciphertext security at all). Finally, our definition of security is stronger than that considered in [31, 26]. Neither work considers the case of an untrusted, physically-secure device. Additionally, [26] prove only that an adversary cannot fully determine an un-exposed key SK_i; we prove the more natural requirement that an adversary cannot break the underlying cryptographic scheme for any (set of) un-exposed time periods.[1]

Our notion of security complements the notion of forward security for digital signatures.[2] This idea, introduced by [3] and subsequently formalized in [5], considers an adversary who compromises the system during a particular time period and thereby obtains *all* the secret information which exists at that point in time. Clearly, in such a setting, one cannot hope to prevent the adversary from signing messages associated with future time periods (since the adversary has all relevant information). Forward-secure signatures, however, prevent the adversary from signing messages associated with *prior* time periods. Many improved constructions of forward-secure signatures have subsequently appeared [1, 24, 21, 27].

Our model uses a stronger assumption, in that we allow for (a limited amount of) physically-secure storage. As a consequence, we are able to obtain a stronger level of security in that the adversary is unable to sign/decrypt messages at *any* non-compromised time period, both in the past and in the future (the latter being impossible for forward-secure schemes). In practice, the type of scheme one uses will depend on the requirements and physical limitations of the

[1] It is easy to design a scheme whose secret keys cannot be fully determined, but which is completely insecure. Furthermore, even if the adversary gets no information about any *single* un-exposed key SK_i, this leaves open the possibility of obtaining joint information about some set of keys, and thereby breaking the scheme in an attack across multiple time periods.

[2] Although forward-security also applies to public-key encryption, forward-secure public-key encryption scheme are not yet known.

system. For many applications, the assumption of (a limited amount of) secure storage is reasonable; it can be realized, for example, by a personal smartcard or a non-networked server.

An identity-based encryption scheme may be converted to an $(N-1, N)$-bidirectionally-secure encryption scheme by viewing each numbered time period as an "identity" and having the physically-secure device implement the trusted third party. Only recently Boneh and Franklin [7] have proposed a practical, identity-based encryption scheme (this was an open problem for many years); they mention the above connection. It should be noted, however, that the security of their scheme is proven in the random oracle model under a very specific, number-theoretic assumption. Additionally, as a minor remark, note that their implementation assumes a trusted server, which our solutions do not. Finally and most importantly, by focusing on the bidirectional-security issue as a basic notion (rather than as an application of identity-based schemes) and concentrating on (t, N)-bidirectional security for $t \ll N$, as we do here, schemes based on alternate assumptions and/or with improved efficiency and functionality may be designed.

2 Definitions

2.1 The Model

In this section, we provide a formal model and definition for bidirectional security. Since the notion may of course be applied to a number of different cryptographic primitives (for example, in Appendix A [32] we discuss bidirectional security of signatures), a generic definition is possible; for simplicity, the definition given here will be specific for the case of public-key encryption. Our definition of a key-updating encryption scheme parallels the definition of a key-evolving signature scheme which appears in [5], with one key difference: in a key-updating scheme there is some data (in particular, SK^*) which is never erased; this data need not be erased since it is stored on a physically-secure device. Furthermore, since the physically-secure device may not be fully trusted, new security concerns arise.

Definition 1 A key-updating (public-key) encryption scheme is a 5-tuple of poly-time algorithms $(\mathcal{G}, \mathcal{U}^*, \mathcal{U}, \mathcal{E}, \mathcal{D})$ such that:

- $\mathcal{G}$, the *key generation algorithm*, is a probabilistic algorithm which takes as input a security parameter 1^k and the total number of time periods N. It returns a public key PK, a master key SK^*, and an initial key SK_0.
- $\mathcal{U}^*$, the *device key-update algorithm*, is a deterministic algorithm which takes as input an index i for a time period (throughout, we assume $1 \leq i \leq N$) and the master key SK^*. It returns (partial) secret key SK'_i for time period i.
- $\mathcal{U}$, the *user key-update algorithm*, is a deterministic algorithm which takes as input an index i, secret key SK_{i-1}, and (partial) secret key SK'_i. It returns secret key SK_i for time period i.
- $\mathcal{E}$, the *encryption algorithm*, is a probabilistic algorithm which takes as input a public-key PK, a time period i, and a message M. It returns a ciphertext $\langle i, C \rangle$.
- $\mathcal{D}$, the *decryption algorithm*, is a deterministic algorithm which takes as input a secret key SK_i and a ciphertext $\langle i, C \rangle$. It returns a message M or the special symbol $\perp$.

For all M in the message space, if $\langle i, C \rangle$ is output by $\mathcal{E}_{PK}(i, M)$, then $\mathcal{D}_{SK_i}(\langle i, C \rangle) = M$. ■

A key-updating encryption scheme is used as one might expect. A user begins by generating $(PK, SK^*, SK_0) \leftarrow \mathcal{G}(1^k, N)$, registering PK in a central location (just as he would for a standard public-key scheme), storing SK^* on a physically-secure device, and storing SK_0 himself.[3] At the beginning of time period i, the user requests $SK'_i = \mathcal{U}^*(i, SK^*)$ from the secure device. Using SK'_i and SK_{i-1}, the user may compute $SK_i = \mathcal{U}(i, SK_{i-1}, SK'_i)$. This key may be used to decrypt messages sent during time period i without further access to the device. After computation of SK_i, the user must erase SK'_i and SK_{i-1}. Note that encryption is always performed with respect to a fixed public key PK which does not need to be changed.

[3]As discussed in the Introduction, the user may store SK_0 himself to guard against compromise of the physically-secure device, or when this device is not trusted. Note that if this device *is* (completely) trusted, we may simply have $SK_0 = \perp$.

Random-Access Key Updates. All the schemes we construct will have a useful property we call *random-access key updates*. For any current period j and any desired period i (either in the future or in the past), it is possible to update the secret key from SK_j to SK_i in "one shot". Namely, we can generalize the key updating algorithms $\mathcal{U}^*$ and $\mathcal{U}$ to take a pair of periods i and j such that $\mathcal{U}^*((i,j), SK^*)$ outputs the partial key SK'_{ij} and $\mathcal{U}((i,j), SK_j, SK'_{ij})$ outputs SK_i. Our definition above corresponds to fixing $j = i - 1$.

2.2 Security

There are two classes of attacks we protect against: (1) repeated compromise of the insecure storage and (2) compromise of the physically-secure device to obtain SK^* (this includes the case when the device itself is untrusted). The first class of attacks includes *key exposures* in which an adversary reads the storage of the insecure device (the *user*) during period i — thereby obtaining secret key SK_i — for up to t time periods of the adversary's choice. We additionally consider *key-update exposures* in which the adversary reads the storage of the user during the key update period between periods $i-1$ and i; in this case, the adversary obtains SK_{i-1}, SK'_i, and SK_i (note that the last of these may be computed from the other two).

To formally model key exposure attacks, we give the adversary access to two (possibly three) types of oracles. The first is a *key exposure oracle* $\mathsf{Exp}_{SK^*,SK_0}(\cdot)$ which, on input i, returns the temporary secret key SK_i (note that SK_i is uniquely defined by SK^* and SK_0). The second is a *left-or-right encryption oracle* (see [4]) $\mathsf{LR}_{PK,\vec{b}}(\cdot,\cdot,\cdot)$, where $\vec{b} = b_1 \dots b_N \in \{0,1\}^N$, which we define by:

$$\mathsf{LR}_{PK,\vec{b}}(i, M_0, M_1) \overset{\text{def}}{=} \mathcal{E}_{PK}(i, M_{b_i}).$$

This models encryption requests by the adversary for time periods and message pairs of his choice. We allow the adversary to interleave encryption requests and key exposure requests, and in particular the key exposure requests of the adversary may be made adaptively and in any order. This is stronger than necessary to model a "real-world" adversary, since, for a real-world adversary, "encryption requests" and "key exposure requests" must occur in chronological order (i.e., after obtaining key SK_i, a real-world adversary could not then decide to obtain SK_{i-1}). However, such attacks might occur in other applications we described (i.e., when using one public key for many employees). Finally, we may also allow the adversary access to a *decryption oracle* $\mathcal{D}^*_{SK^*,SK_0}(\cdot)$ which, on input $\langle i, C\rangle$, computes $\mathcal{D}_{SK_i}(\langle i, C\rangle)$. This models a chosen-ciphertext attack by the adversary.

The vector $\vec{b}$ for the left-or-right oracle will be chosen randomly, and the adversary succeeds by guessing the value of b_i for any un-exposed time period i. Informally, a scheme is secure if any PPT adversary has success negligibly close to $1/2$. More formally:

Definition 2 Let $\Pi = (\mathcal{G}, \mathcal{U}^*, \mathcal{U}, \mathcal{E}, \mathcal{D})$ be a key-updating encryption scheme. For adversary A, define the following:

$$\begin{aligned}\mathsf{Succ}_{A,\Pi}(k) \quad \overset{\text{def}}{=} \quad \Pr\Big[&(PK, SK^*, SK_0) \leftarrow \mathcal{G}(1^k, N); \vec{b} \leftarrow \{0,1\}^N; \\ &(i, b') \leftarrow A^{\mathsf{LR}_{PK,\vec{b}}(\cdot,\cdot,\cdot), \mathsf{Exp}_{SK^*,SK_0}(\cdot), \mathcal{O}(\cdot)}(PK) : b' = b_i\Big],\end{aligned}$$

where i was never submitted to $\mathsf{Exp}_{SK^*,SK_0}(\cdot)$, and $\mathcal{O}(\cdot) = \perp$ for a plaintext-only attack and $\mathcal{O}(\cdot) = \mathcal{D}^*_{SK^*,SK_0}(\cdot)$ for a chosen-ciphertext attack (in the latter case the adversary is not allowed to query $\mathcal{D}^*(\langle i, C\rangle)$ if $\langle i, C\rangle$ was returned by $\mathsf{LR}(i,\cdot,\cdot)$). Π is (t, N)-*bidirectionally-secure* if, for any PPT A who submits at most t requests to the key-exposure oracle, $|\mathsf{Succ}_{A,\Pi}(k) - 1/2|$ is negligible. ■

As mentioned above, we may also consider attacks in which an adversary breaks in to the user's storage while a key update is taking place (i.e., the exposure occurs between two periods $i-1$ and i); we call this a *key-update exposure* at period i. In this case, the adversary receives SK_{i-1}, SK'_i, and (can compute) SK_i. Informally, we say a scheme has *secure key updates* if a key-update exposure at period i is equivalent to key exposures at periods $i-1$ and i and no more. More formally:

Definition 3 Key-updating encryption scheme Π has *secure key updates* if the view of any adversary A making a key-update exposure request at period i can be perfectly simulated by an adversary A' who makes key exposure requests at periods $i-1$ and i. ■

This property is desirable in real-world implementations of a key-updating encryption scheme since an adversary who gains access to the user's storage is likely to have access for several consecutive time periods (i.e., until the user detects or re-boots), including the key updating steps.

We also consider attacks which compromise the physically-secure device (this includes attacks in which this device is untrusted). Here, our definition requires that the encryption scheme be secure against an adversary which is given SK^* as input. Note that we do *not* require security against an adversary who compromises both the user's storage and the secure device — in our model this is impossible since, given SK^* and SK_i, an adversary can compute SK_j (at least for $j > i$) by himself.

Definition 4 Let Π be a key-updating scheme which is (t, N)-bidirectionally-secure. For any adversary B, define the following:

$$\begin{aligned}\mathsf{Succ}_{B,\Pi}(k) \quad \stackrel{\text{def}}{=} \quad & \Pr\Big[(PK, SK^*, SK_0) \leftarrow \mathcal{G}(1^k, N); \vec{b} \leftarrow \{0,1\}^N; \\ & \qquad (i, b') \leftarrow B^{\mathsf{LR}_{PK,\vec{b}}(\cdot,\cdot,\cdot),\mathcal{O}(\cdot)}(PK, SK^*) : b' = b_i\Big],\end{aligned}$$

where $\mathcal{O}(\cdot) = \perp$ for a plaintext-only attack and $\mathcal{O}(\cdot) = \mathcal{D}^*_{SK^*,SK_0}(\cdot)$ for a chosen-ciphertext attack (in the latter case the adversary is not allowed to query $\mathcal{D}^*(\langle i, C\rangle)$ if $\langle i, C\rangle$ was returned by $\mathsf{LR}(i, \cdot, \cdot)$). Π is *strongly* (t, N)-bidirectionally-secure if, for any PPT B, $|\mathsf{Succ}_{B,\Pi}(k) - 1/2|$ is negligible. ■

3 Our Constructions

3.1 Generic Semantically-Secure Construction

Let (G, E, D) be any semantically secure encryption scheme. Rather than giving a separate (by now, standard) definition, we may view it as the special case of a $(0, 1)$-bidirectionally-secure encryption scheme. Namely, only one secret key SK is present, and any PPT adversary, given PK and the left-or-right-oracle $\mathsf{LR}_{PK,b}$ for an unknown bit b, cannot predict b with success non-negligibly different from $1/2$. Hence, our construction below can be viewed as an amplification of a $(0, 1)$-bidirectionally-secure scheme into a general (t, N)-bidirectionally secure scheme.

In the following, we will assume that $t, \log N = O(\mathsf{poly}(k))$, where k is our security parameter. Thus, we allow exponentially-many periods, and can tolerate exposure of any polynomial number of keys. For concreteness, we assume that E operates on messages of length $\ell = \ell(k)$, and we construct a (t, N)-bidirectionally secure scheme operating on messages of length $L = L(k)$.

Auxiliary Definitions. We need two auxiliary definitions: that of an *all-or-nothing transform* [29, 8] (AONT) and a *cover-free family* [14, 12]. Informally, an AONT splits the message M into n secret shares $x_1, \ldots, x_n$ (and possibly one public share z), and has the property that (1) the message M can be efficiently recovered from *all* the shares $x_1, \ldots, x_n, z$, but (2) missing even a single share x_j gives "no information" about M.[4] We formalize this, modifying the conventional definitions [29, 8, 9] to a form more compatible with our prior notation.

Definition 5 An efficient randomized transformation $\mathcal{T}$ is called an (L, ℓ, n)-AONT (where L, ℓ, n are polynomial functions of the security parameter k) if: (1) on input $M \in \{0,1\}^L$, $\mathcal{T}$ outputs $(X, z) \stackrel{\text{def}}{=} (x_1, \ldots, x_n, z)$, where $x_j \in \{0,1\}^\ell$; (2) there exists an efficient inverse function $\mathcal{I}$ such that $\mathcal{I}(X, z) = M$; (3) $\mathcal{T}$ satisfies the indistinguishability property described below.

Let $X_{-j} = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$, and define $\mathcal{T}_{-j}(M) = (X_{-j}, z)$, where $(X, z) \leftarrow \mathcal{T}(M)$. We define the left-or-right oracle $\mathsf{LR}_b(j, M_0, M_1) \stackrel{\text{def}}{=} \mathcal{T}_{-j}(M_b)$, where $b \in \{0,1\}$. Now, for a PPT adversary A, we let $\mathsf{Succ}_{A,\mathcal{T}}(k) \stackrel{\text{def}}{=} \Pr[b \leftarrow \{0,1\}; b' \leftarrow A^{\mathsf{LR}_b(\cdot,\cdot,\cdot)}(1^k) \ : \ b' = b]$. We require that $|\mathsf{Succ}_{A,\mathcal{T}}(k) - 1/2|$ is negligible. ■

A family of subsets $S_1 \ldots S_N$ over some universe U is said to be *t-cover-free* if no t subsets $S_{i_1}, \ldots, S_{i_t}$ contain a (different) subset S_i; in other words, for all $\{i_0, \ldots, i_t\}$ with $i_0 \notin \{i_1, \ldots, i_t\}$, we have $S_{i_0} \not\subseteq \cup_{j=1}^{t} S_{i_j}$. A family is

[4] This is a generalization of $(n-1, n)$-secret sharing, except it permits a public share and computational secrecy.

said to be (t,α)*-cover-free*, where $0 < \alpha < 1$, if, for all $\{i_0,\ldots,i_t\}$ with $i_0 \notin \{i_1,\ldots,i_t\}$, we have $|S_{i_0} \setminus \cup_{j=1}^{t} S_{i_j}| \geq \alpha|S_i|$. Such families are well known and have been used several times in cryptographic applications [10, 25, 16]. In what follows, we fix $\alpha = 1/2$ for simplicity, and will use the following (essentially optimal) result, non-constructively proven by [14], and subsequently made efficient by [25, 20].

Theorem 1 ([14, 25, 20]) *For any N and t, one can efficiently construct a $(t,\frac{1}{2})$-cover-free collection of N subsets $S_1,\ldots,S_N$ of $U = [u]$, with $|S_i| = n$ for all i, satisfying: $u = \Theta(t^2 \log N)$ and $n = \Theta(t \log N)$.*

In particular, since we assumed that $t, \log N = O(\mathsf{poly}(k))$, we have $u, n = O(\mathsf{poly}(k))$ as well.

Construction. For simplicity, we first describe the scheme which is not strongly secure (see Definition 4), and then show a modification making it strongly secure. Let $S_1,\ldots,S_N \subset [u]$ be the $(t,\frac{1}{2})$-cover-free family of n-element sets, as given by Theorem 1. Also, let $\mathcal{T}$ be a secure (L,ℓ,n)-AONT. Our (t,N)-bidirectionally secure scheme will have a set of u independent encryption/decryption keys (sk_r, pk_r) for our basic encryption E, of which only the subset S_i will be used at time period i. Specifically, the public key of the scheme will be $PK = \{pk_1,\ldots,pk_u\}$, the secret key at time i will be $SK_i = \{sk_r \ : \ r \in S_i\}$, and the master key (for now) will be $SK^* = \{sk_1,\ldots,sk_u\}$. We now define the encryption of $M \in \{0,1\}^L$ at time period i as:

$$\mathcal{E}_{PK}(i,M) = \langle\ i,\ (E_{pk_{r_1}}(x_1),\ldots,E_{pk_{r_n}}(x_n),\ z)\ \rangle,$$

where $(x_1,\ldots,x_n,z) \leftarrow \mathcal{T}(M)$ and $S_i = \{r_1,\ldots,r_n\}$. To decrypt $\langle i,(y_1,\ldots,y_n,z)\rangle$ using $SK_i = \{sk_r \ : \ r \in S_i\}$, the user first recovers the x_j's from the y_j's using D, and then recovers the message $M = \mathcal{I}(x_1,\ldots,x_n,z)$. Key updates take place the expected way: at the beginning of period i, the device simply sends the user its new secret key SK_i, and the user erases its old key SK_{i-1}.

We remark that this scheme obviously supports secure key updates, as well as random-access key updates.

Security. We informally argue the (t,N)-bidirectional security of this scheme. The definition of the AONT tells us that the system is secure at time period i provided the adversary misses *at least one* key sk_r, where $r \in S_i$. Indeed, semantic security of E implies that the adversary completely misses the shares encrypted with sk_r in this case, and hence has no information about the message M. On the other hand, if the adversary learn any t keys $SK_{i_1},\ldots,SK_{i_t}$, he learns the auxiliary keys $\{sk_r \ : \ r \in S_{i_1} \cup S_{i_2} \ldots \cup S_{i_t}\}$. Hence, the necessary and sufficient condition for (t,N)-bidirectional security is exactly the t-cover freeness of the S_i's! The parameter $\alpha = \frac{1}{2}$ is used to improve the security of our reduction.

Theorem 2 *Let (G,E,D) be a semantically-secure encryption scheme, $\mathcal{T}$ a secure (L,ℓ,n)-AONT, and $S_1,\ldots,S_N$ a $(t,\frac{1}{2})$-cover-free collection, where $L,\ell,n,t,u,\log N = O(\mathsf{poly}(k)$ (k is the security parameter). Then the generic scheme Π described above is (t,N)-bidirectionally secure. Moreover, if one can break the indistinguishability of Π with advantage ε, then one can do the same for either (G,E,D) or $\mathcal{T}$ with advantage at least $\Omega(\varepsilon/t)$.*

Proof: Let A be the adversary for Π with $\mathsf{Succ}_{A,\Pi}(k) = \frac{1}{2}+\varepsilon$. First, we create the following adversary A' such that $\mathsf{Succ}_{A',\Pi} \geq \frac{1}{2}+\varepsilon\cdot\frac{\alpha n}{u} = \frac{1}{2}+\Omega(\frac{\varepsilon}{t})$. A' first picks a random index $r \in [u]$. Then it runs A up to the point when A outputs (i,b'). At this stage, A' looks at indices $i_1,\ldots,i_t$ of the t exposed time periods, and checks if $r \in S_i \setminus \cup_{j=1}^{t} S_{i_j}$. If this test succeeds, A' also outputs (i,b'). Else, it outputs (i,c), where c is a random bit. In other words, A' uses the output of A provided the guess r is such that sk_r is used at period i, but A did not learn sk_r. Notice, since A cannot output $i \in \{i_1 \ldots i_t\}$ and since our family is $(t,\frac{1}{2})$-cover-free, there are at least $\alpha|S_i| = n/2$ indices $r' \in S_i \setminus \cup_{j=1}^{t} S_{i_j}$. Also, since A' chose $r \in [u]$ at random and independently of the run of A, with probability at least $q = \frac{n}{2u} = \Omega(\frac{1}{t})$ we get that A' will use the output of A, so that $\mathsf{Succ}_{A',\Pi} \geq (1-q)\frac{1}{2}+q(\frac{1}{2}+\varepsilon) \geq \frac{1}{2}+\Omega(\frac{\varepsilon}{t})$, as claimed.

Next, we create a more favorable environment for this A' to simplify the proof. Right after A' picks its random r, we give A' all the secret keys sk_p for all $p \neq r$. At this point, there is no need to encrypt with any of the keys other than pk_r (A' can decrypt anyway). Moreover, there is no need for our environment to pick a full-fledged N-bit vector $\vec{b}$. Precisely, only b_i's such that $r \in S_i$ should be chosen. In fact, we make the life of A' even easier. Rather than choosing the b_i's (where $r \in S_i$) independently from each other, we choose only one random bit b, and set all $b_i = b$, for all i s.t. $r \in S_i$. Clearly, this only helps A'.[5] Notice, since A' is committed to output a non-random bit b' only for

[5]One way to formally see this is to think that we picked all the b_i's independently, then picked a random b and then told A' the set of i such that $b_i = b$ (and thus, the set of i where $b_i = 1-b$), but did not tell b.

period i such that $r \in S_i$ and the original adversary A did not learn sk_r, we get that $\Pr(b' = b) \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$ in the modified environment.

To summarize, we can assume A' runs in the following environment Env_0. A' picks a random $r \in [u]$. We pick a random key pair (sk_r, pk_r) for E and a random bit $b \in \{0,1\}$. We give A' the public key pk_r, and access to the "reduced" left-or-right oracle $\mathsf{LR}'_{pk_r,b}(i, M_0, M_1)$ which can be called only for i satisfying $r \in S_i$. The oracle runs $(X, z) \leftarrow \mathcal{T}(M_b)$, and returns the following: $(T_{-j}(M_b), E_{pk_r}(x_j))$, where $j \in [n]$ is the position of r inside S_i. The goal of A' is to predict b, and we assumed that it does so correctly with probability $q_0 = \Pr(b = b' \mid Env_0) \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$.

Next, we run A' in a different environment Env_1. It is identical to Env_0 except that on left-or-right query (i, M_0, M_1) (where $r \in S_i$), rather than returning $(T_{-j}(M_b), E_{pk_r}(x_j))$, Env_1 instead returns $(T_{-j}(M_b), E_{pk_r}(0))$. Namely, it encrypts the all-zero string 0 instead of the share x_j. We let $q_1 = \Pr(b = b' \mid Env_1)$.

The proof is now almost complete. The fact that $q_0 \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$ implies that either: (a) $q_0 - q_1 \geq \Omega(\frac{\varepsilon}{t})$; or (b) $q_1 \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$. We show that either case is a contradiction: case (a) to the indistinguishability of encryption E, while case (b) to the indistinguishability of AONT $\mathcal{T}$.

Case (a): If $q_0 - q_1 \geq \Omega(\frac{\varepsilon}{t})$, we break the indistiguishability of E by means of the following adversary A_1 which in turn runs A' as follows. When A' chooses $r \in [u]$, A_1 views the public key of E as pk_r and picks a random $b \in \{0,1\}$. From now on, A_1 runs A' and answers the left-or-right queries (i, M_0, M_1) of A' as follows. If $r \notin S_i$, it ignores it. Else, it sets $(X, z) \leftarrow \mathcal{T}(M_b)$, and gives its *own left-or-right oracle* the query $(x_j, 0)$, where j is the position of r inside S_i. When it gets y (encryption of either x_j or 0) back from its oracle, it returns to A' the answer (X_{-j}, z, y). When A' finally outputs its guess b', A_1 checks if $b = b'$. If so, it guesses its own bit d was 0 (i.e., x_j was always encrypted), else that it was 1 (0 was always encrypted). It is easy to see that if $d = 0$, we exactly run A' in Env_0, else — exactly in Env_1. Hence, A_1 predicts d correctly with probability $\frac{1}{2}(1 - q_1) + \frac{1}{2}q_0 \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$, contradicting the security of E.

Case (b): If $q_1 \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$, we break the indistingushability of $\mathcal{T}$ by means of the following adversary A_2 which in turn runs A' as follows. A_2 picks a random key (pk_r, sk_r) and runs A' up to completion, outputting the same b' as A' does. To answer the left-or-right-query (i, M_0, M_1), where $r \in S_i$, A_2 calls its own oracle of (j, M_0, M_1), where j is the position of r inside S_i. It gets back $T_{-j}(M_b)$, and returns A' the pair $(T_{-j}(M_b), E_{pk_r}(0))$. Clearly, A_2 exactly recreates Env_1, and hence predicts its own b with probability $q_1 \geq \frac{1}{2} + \Omega(\frac{\varepsilon}{t})$, contradicting security of $\mathcal{T}$. ■

Strong Bidirectional Security. As described, the scheme above is not *strongly* (t, N)-bidirectionally secure since the device stores all the secret keys $(sk_1, \ldots, sk_u)$. However, we can easily make it such. Namely, the user picks one extra key pair (sk_0, pk_0). It publishes pk_0 together with the other public keys, but keeps sk_0 to itself (never erasing it). We assume that $\mathcal{T}$ now produces $(n+1)$ secret shares $x_0, \ldots, x_n$ rather than n, and the first share x_0 is always encrypted with sk_0 (while the others, as before, with the corresponding keys inside S_i). Formally, let $S'_i = S_i \cup \{0\}$, the master key is still $SK^* = \{sk_1, \ldots, sk_u\}$, but now $PK = \{pk_0, pk_1, \ldots, pk_u\}$ and the i-th secret key is $SK_i = \{sk_r \,:\, r \in S'_i\}$. Strong bidirectional security follows the same proof as Theorem 2. In fact, we do not even lose a factor t in the security, since we do not pick r at random, but always set it to 0.

Efficiency. The main parameters of the scheme are: (1) the size of PK and SK^* are both $u = O(t^2 \log N)$; and (2) the user storage and the number of local encryptions per global encryption are both $n = O(t \log N)$. In particular, the surprising aspect of our construction is that it supports an *exponential* number of periods N, and the main parameters depend mainly on t, the number of exposures we allow. Since t is usually quite small (say, $t = O(1)$ and certainly $t \ll N$), we obtain good parameters considering the generality of the scheme. (In Section 3.2 we use a specific encryption scheme and achieve $|PK|, |SK^*| = O(t)$ and $|SK_i| = O(1)$.)

Additionally, the choice of a secure (L, ℓ, n)-AONT defines the tradeoff between the number of encrypted bits L compared to the total encryption size, which is $(\beta n \ell + |z|)$, where β is the expansion of E, and $|z|$ is the size of the public share. In particular, if $L = \ell$, we can use any traditional $(n-1, n)$-secret sharing scheme (e.g., Shamir's scheme [30], or even XOR-sharing: pick random x_j's subject to $M = \bigoplus x_j$). This way we have no public part, but the ciphertext increases by a factor of βn as compared to the plaintext. Computationally-secure AONT's allow for better tradeoffs. For example, using either the computational secret sharing scheme of [23], or the AONT constructions of [9], we can achieve $|z| = L$, while ℓ can be as small as the security parameter k (in particular, $\ell \ll L$). Thus, we get *additive* increase $\beta n\ell$, which is essentially independent of L. Finally, in the random oracle model, we could use the construction of [8] achieving $|z| = 0$, $L = \ell(n-1)$, so the ciphertext size is $\beta\ell n \approx \beta L$.

Adaptive vs. Non-adaptive adversaries. Theorem 2 holds for an *adaptive* adversary who makes key exposure requests based on all information collected so far. We notice, however, that both the security and the efficiency of our construction could be somewhat improved for non-adaptive adversaries, who choose the key-exposure periods $i_1, \ldots, i_t$ at the outset of the protocol (which is the model of [31, 26, 2]). For example, it is easy to see that we no longer lose the factor t in the security of our reduction in Theorem 2 (instead, we only loose a small constant factor). As for the efficiency, instead of using an AONT (which is essentially an $(n-1, n)$-secret sharing scheme), we can now use any $(n/2, n)$-"ramp" secret sharing scheme [6]. This means that n shares reconstruct the secret, but any $n/2$ shares yield no information about the secret. Indeed, since our family is $(t, \frac{1}{2})$-cover-free, any non-exposed period will have the adversary miss more than half of the relevant secret keys. For non-adaptive adversaries, we know at the outset which secret keys are non-exposed, and can use a simple hybrid argument over these keys to prove the security of the modified scheme. For example, we can use the "ramp" generalization of Shamir's secret sharing scheme[6] proposed by Franklin and Yung [15], and achieve $L = \ell n/2$ instead of $L = \ell$ resulting from regular Shamir's $(n-1, n)$-scheme.

3.2 Semantic Security Based on DDH

In this section, we present an efficient strongly (t, N)-bidirectionally secure scheme, whose semantic security can be proved under the DDH assumption. Section 3.3 discusses modifications to the scheme which achieve chosen-ciphertext security.

We first describe the basic encryption scheme we build upon. The key generation algorithm $\mathsf{Gen}(1^k)$ selects a random prime q with $|q| = k$ such that $p = 2q + 1$ is prime. This defines a unique subgroup $\mathbb{G} \subset \mathbb{Z}_p^*$ of size q in which the DDH assumption is assumed to hold; namely, it is hard to disinguish a random tuple (g, h, u, v) of four independent elements in $\mathbb{G}$ from a random tuple satisfying $\log_g u = \log_h v$. Given group $\mathbb{G}$, key generation proceeds by selecting random elements $g, h \in \mathbb{G}$ and random $x, y \in \mathbb{Z}_q$. The public key consists of g, h, and the Pedersen commitment [28] to x and y: $z = g^x h^y$. The secret key contains both x and y. To encrypt $M \in \mathbb{G}$, choose random $r \in \mathbb{Z}_q$ and compute $(g^r, h^r, z^r M)$. To decrypt (u, v, w), compute $M = w/u^x v^y$. This scheme is very similar to El Gamal encryption [13], except it uses two generators. It has been recently used by [22] in a different context.

Our (t, N)-bidirectional scheme builds on the above basic encryption scheme and is presented in Figure 1. The key difference is that, after choosing $\mathbb{G}, g, h$, as above, we select two random polynomials $f_x(\tau) \stackrel{\text{def}}{=} \sum_{j=0}^{t} x_j^* \tau^j$ and $f_y(\tau) \stackrel{\text{def}}{=} \sum_{j=0}^{t} y_j^* \tau^j$ over $\mathbb{Z}_q$ of degree t. The public key consists of g, h and Pedersen commitments $\{z_0^*, \ldots, z_t^*\}$ to the coefficients of the two polynomials (see Figure 1). The user stores the constant terms of the two polynomials (i.e., x_0^* and y_0^*) and the remaining coefficients are stored by the physically-secure device.

To encrypt during period i, first z_i is computed from the public key as $z_i \stackrel{\text{def}}{=} \Pi_{j=0}^{t} (z_j^*)^{i^j}$. Then (similar to the basic scheme), encryption of message M is done by choosing $r \in \mathbb{Z}_q$ at random and computing $(g^r, h^r, z_i^r M)$. Using our notation from above, it is clear that $z_i = g^{f_x(i)} h^{f_y(i)}$. Thus, as long as the user has secret key $SK_i = (f_x(i), f_y(i))$ during period i, decryption during that period may be done just as in the basic scheme. In fact, this is the case. The user begins with $SK_0 = (x_0^*, y_0^*) = (f_x(0), f_y(0))$. At the start of any period i, when the user needs to update his key, the device transmits partial key $SK_i' = (x_i', y_i')$ to the user. Note that (cf. Figure 1) $x_i' = f_x(i) - f_x(i-1)$ and $y_i' = f_y(i) - f_y(i-1)$. Thus, since the user already has SK_{i-1}, the user may easily compute SK_i from these values. At this point, the user erases SK_{i-1}, and uses SK_i to decrypt for the remainder of the time period.

Theorem 3 *Under the DDH assumption, the encryption scheme of Figure 1 is strongly (t, N)-bidirectionally secure under plaintext-only attacks. Furthermore, it has secure key updates and supports random-access key updates.*

Proof: That the scheme has secure key updates is trivial to show, since an adversary who exposes keys SK_{i-1} and SK_i via key exposure requests can compute the value SK_i' itself and thereby perfectly simulate a key-update exposure at period i. Similarly, random-access key updates can be done using partial keys $SK_{ij}' = (x_{ij}', y_{ij}')$, where

[6]Here the message length $L = \ell n/2$, and the ℓ-bit parts $m_1, \ldots, m_{n/2}$ of M are viewed as the $n/2$ lower order coefficients of an otherwise random polynomial of degree $(n-1)$ over $GF[2^\ell]$. This polynomial is then evaluated at n points of $GF[2^\ell]$ to give the final n shares.

```
$\mathcal{G}(1^k)$
    $(g, h, q) \leftarrow \mathsf{Gen}(1^k)$;  $x_0^*, y_0^*, \ldots, x_t^*, y_t^* \leftarrow \mathbb{Z}_q$
    $z_0^* := g^{x_0^*} h^{y_0^*}, \ldots, z_t^* := g^{x_t^*} h^{y_t^*}$;  $PK := (g, h, z_0^*, \ldots, z_t^*)$
    $SK^* := (x_1^*, y_1^*, \ldots, x_t^*, y_t^*)$;  $SK_0 := (x_0^*, y_0^*)$
    return $PK, SK^*, SK_0$
```

```
$\mathcal{U}^*(i, SK^* = (x_1^*, y_1^*, \ldots, x_t^*, y_t^*))$
    $x_i' := \sum_{j=1}^{t} x_j^* \left(i^j - (i-1)^j\right)$
    $y_i' := \sum_{j=1}^{t} y_j^* \left(i^j - (i-1)^j\right)$
    return $SK_i' = (x_i', y_i')$
```

```
$\mathcal{U}(i, SK_{i-1} = (x_{i-1}, y_{i-1}), SK_i' = (x_i', y_i'))$
    $x_i := x_{i-1} + x_i'$
    $y_i := y_{i-1} + y_i'$
    return $SK_i = (x_i, y_i)$
```

```
$\mathcal{E}_{(g,h,z_0^*,\ldots,z_y^*)}(i, M)$
    $z_i := \Pi_{j=0}^{t} (z_j^*)^{i^j}$
    $r \leftarrow \mathbb{Z}_q$
    $C := (g^r, h^r, z_i^r M)$
    return $\langle i, C \rangle$
```

```
$\mathcal{D}_{(x_i,y_i)}(\langle i, C = (u, v, w) \rangle)$
    $M := w / u^{x_i} v^{y_i}$
    return $M$
```

Figure 1: Semantically-secure key-updating encryption scheme based on DDH.

$x_{ij}' = f_x(i) - f_x(j)$, $y_{ij}' = f_y(i) - f_y(j)$. The user then recovers $SK_i = (x_i, y_i)$ from $SK_j = (x_j, y_j)$ and SK_{ij}' via $x_i = x_j + x_{ij}'$, $y_i = y_j + y_{ij}'$.

We now show that the scheme satisfies Definition 2. By a standard hybrid argument [4], it is sufficient to consider an adversary A who asks a single query to its left-or-right oracle (for time period i of the adversary's choice) and must guess the value b_i.

Assume A makes only a single query to the LR oracle during period i for which it did not make a key exposure request. In the original experiment (cf. Figure 1), the output of $\mathsf{LR}_{PK,\vec{b}}(i, M_0, M_1)$ is defined as follows: choose $r \in \mathbb{Z}_q$ at random and output $(g^r, h^r, z_i^r M_{b_i})$. Given a tuple (g, h, u, v) which is either a DDH tuple or a random tuple, modify the original experiment as follows: the output of $\mathsf{LR}_{PK,\vec{b}}(i, M_0, M_1)$ will be $(u, v, u^{x_i} v^{y_i} M_b)$. Note that if (g, h, u, v) is a DDH tuple, then this is a perfect simulation of the original experiment. On the other hand, if (g, h, u, v) is a random tuple then, under the DDH assumption, the success of any PPT adversary in this modified experiment cannot differ by more than a negligible amount from its success in the original experiment. It is important to note that, in running the experiment, we can answer all of A's key exposure requests correctly since all secret keys are known. Thus, in contrast to [31, 26], we may handle an adaptive adversary who chooses when to make key exposure requests based on all information seen during the experiment.

Assume now that (g, h, u, v) is a random tuple and $\log_g h \neq \log_u v$ (this will occur with all but negligible probability). We claim that the adversary's view in the modified experiment is independent of $\vec{b}$. Indeed, the adversary knows only $t-1$ values of $f_x(\cdot)$ and $f_y(\cdot)$ (at points other than i), and since both $f_x(\cdot)$ and $f_y(\cdot)$ are random polynomials of degree t, the values x_i, y_i ($= f_x(i), f_y(i)$) are *information-theoretically* uniformly distributed, subject only to:

$$\log_g z_i = x_i + y_i \log_g h. \tag{1}$$

Consider the output of the encryption oracle $(u, v, u^{x_i} v^{y_i} M_b)$. Since:

$$\log_u (u^{x_i} v^{y_i}) = x_i + y_i \log_u v, \tag{2}$$

and (1) and (2) are linearly independent, the conditional distribution of $u^{x_i} v^{y_i}$ (conditioned on b_i and the adversary's view) is uniform. Thus, the adversary's view is independent of b_i (and hence $\vec{b}$). This implies that the success probability of A in this modified experiment is $1/2$, and hence the success probability of A in the original experiment is at most negligibly different from $1/2$.

We now consider security against compromises of the physically-secure device; in this case, there are no key exposure requests but the adversary learns SK^*. Again, it is sufficient to consider an adversary who asks a single query to its left-or-right oracle (for time period i of the adversary's choice) and must guess the value b_i. Since SK^* only contains the t highest-order coefficients of t-degree polynomials, the pair (x_i, y_i) is information-theoretically uniformly distributed (for all i) subject to $x_i + y_i \log_g h = \log_g z_i$. An argument similar to that given previously shows that the success probability of the adversary is at most negligibly better than $1/2$, and hence the scheme satisfies Definition 4. ■

3.3 Chosen-Ciphertext Security Based on DDH

We may modify the scheme given in the previous section so as to be resistant to chosen-ciphertext attacks. In doing so, we build upon the chosen-ciphertext-secure (standard) public-key encryption scheme of Cramer and Shoup [11].

```
$\mathcal{G}(1^k)$
    $(g, h, q) \leftarrow \mathsf{Gen}(1^k)$; $H \leftarrow \mathsf{UOWH}(1^k)$
    for $i = 0$ to $t$ and $n = 0$ to 2:
        $x^*_{i,n}, y^*_{i,n} \leftarrow \mathbb{Z}_q$
    for $i = 0$ to $t$:
        $z^*_i := g^{x^*_{i,0}} h^{y^*_{i,0}}$; $c^*_i := g^{x^*_{i,1}} h^{y^*_{i,1}}$; $d^*_i := g^{x^*_{i,2}} h^{y^*_{i,2}}$
    $PK := (g, h, H, \{z^*_i, c^*_i, d^*_i\}_{0 \le i \le t})$
    $SK^* := (\{x^*_{i,n}, y^*_{i,n}\}_{2 \le i \le t,\, 0 \le n \le 2})$; $SK_0 := (\{x^*_{i,n}, y^*_{i,n}\}_{0 \le i \le 1,\, 0 \le n \le 2})$
    return $PK, SK^*, SK_0$
```

```
$\mathcal{U}^*(i, SK^*)$
    for $n = 0$ to 2:
        $x'_{i,n} := \sum_{j=2}^{t} x^*_{j,n} \left(i^j - (i-1)^j\right)$
        $y'_{i,n} := \sum_{j=2}^{t} y^*_{j,n} \left(i^j - (i-1)^j\right)$
    return $SK'_i = (\{x'_{i,n}, y'_{i,n}\}_{0 \le n \le 2})$
```

```
$\mathcal{U}(i, SK_{i-1}, SK'_i)$
    for $n = 0$ to 2:
        $x_{i,n} = x_{i-1,n} + x'_{i,n} + x_{1,n}$
        $y_{i,n} = y_{i-1,n} + y'_{i,n} + y_{1,n}$
    return $SK_i = (\{x_{i,n}, y_{i,n}, x_{1,n}, y_{1,n}\}_{0 \le n \le 2})$
```

```
$\mathcal{E}_{PK}(i, M)$
    $z_i := \Pi_{j=0}^{t} (z^*_j)^{i^j}$; $c_i := \Pi_{j=0}^{t} (c^*_j)^{i^j}$
    $d_i := \Pi_{j=0}^{t} (d^*_j)^{i^j}$
    $r \leftarrow \mathbb{Z}_q$
    $C := (g^r, h^r, z_i^r M, (c_i d_i^\alpha)^r)$,
        where $\alpha \overset{\text{def}}{=} H(g^r, h^r, z_i^r M)$
    return $\langle i, C \rangle$
```

```
$\mathcal{D}_{SK_i}(\langle i, C = (u, v, w, e) \rangle)$
    $\alpha := H(u, v, w)$
    if $u^{x_{i,1} + x_{i,2}\alpha} v^{y_{i,1} + y_{i,2}\alpha} \neq e$
        return $\perp$
    else $M := w / u^{x_{i,0}} v^{y_{i,0}}$
    return $M$
```

Figure 2: Chosen-ciphertext-secure key-updating encryption scheme based on DDH.

We briefly review the "basic" Cramer-Shoup scheme (in part to conform to the notation used in Figure 2). Given generators g, h of group $\mathbb{G}$ (as described in the previous section), secret keys $\{x_n, y_n\}_{0 \le n \le 2}$ are chosen randomly from $\mathbb{Z}_q$. Then, public-key components $z = g^{x_0} h^{y_0}$, $c = g^{x_1} h^{y_1}$, and $d = g^{x_2} h^{y_2}$ are computed. In addition, a function H is randomly chosen from a family of universal one-way hash functions. The public key is (g, h, z, c, d, H).

To encrypt a message $M \in \mathbb{G}$, a random element $r \in \mathbb{Z}_q$ is chosen and the ciphertext is: $(g^r, h^r, z^r M, (cd^\alpha)^r)$, where $\alpha \overset{\text{def}}{=} H(g^r, h^r, z^r M)$. To decrypt a ciphertext (u, v, w, e), first check whether $u^{x_1 + x_2\alpha} v^{y_1 + y_2\alpha} = e$. If not, output $\perp$. Otherwise, output $M = w / u^{x_0} v^{y_0}$.

In our extended scheme (cf. Figure 2), we choose six random, degree-t polynomials (over $\mathbb{Z}_q$) f_{x_0}, f_{y_0}, f_{x_1}, f_{y_1}, f_{x_2}, f_{y_2}, where $f_{x_n}(\tau) \overset{\text{def}}{=} \sum_{j=0}^{t} x^*_{j,n} \tau^j$ and $f_{y_n}(\tau) \overset{\text{def}}{=} \sum_{j=0}^{t} y^*_{j,n} \tau^j$ for $0 \le n \le 2$. The public key consists of g, h, H, and Pedersen commitments to the coefficients of these polynomials. The user stores the constant term *and* the coefficient of the linear term for each of these polynomials, and the remaining coefficients are stored by the physically-secure device.

To encrypt during period i, a user first computes z_i, c_i, and d_i by evaluating the polynomials "in the exponent" (see Figure 2). Then, just as in the basic scheme, encryption of M is performed by choosing random $r \in \mathbb{Z}_q$ and computing $(g^r, h^r, z_i^r M, (c_i d_i^\alpha)^r)$, where $\alpha \overset{\text{def}}{=} H(g^r, h^r, z_i^r M)$. Notice that $z_i = g^{f_{x_0}(i)} h^{f_{y_0}(i)}$, $c_i = g^{f_{x_1}(i)} h^{f_{y_1}(i)}$, and $d_i = g^{f_{x_2}(i)} h^{f_{y_2}(i)}$. Thus, the user can decrypt (just as in the basic scheme) as long as he has $f_{x_n}(i), f_{y_n}(i)$ for

$0 \leq n \leq 2$. In fact, the secret key SK_i includes these values; in addition, the secret key at all times includes the linear coefficients $x^*_{1,0}, y^*_{1,0}, \ldots, x^*_{1,2}, y^*_{1,2}$. These values are used to help update SK_i.

Theorem 4 *Under the DDH assumption, the encryption scheme of Figure 2 is strongly $(t-2, N)$-bidirectionally secure under chosen-ciphertext attacks. Furthermore, the scheme has secure key updates and supports random-access key updates.*

Proof: That the scheme has secure key updates is trivial, since SK'_i may be computed from SK_{i-1} and SK_i. Random-access key updates are done analogously to the scheme of the previous section. We now show the bidirectional security of the scheme (cf. Definition 2). A standard hybrid argument [4] shows that it is sufficient to consider an adversary A who makes only a single request to its left-or-right oracle (for time period i of the adversary's choice) and must guess the value b_i. We stress that polynomially-many calls to the decryption oracle are allowed.

Assume A makes a single query to the LR oracle during period i for which it did not make a ley exposure request. In the original experiment (cf. Figure 2), the output of $\mathsf{LR}_{PK,\vec{b}}(i, M_0, M_1)$ is as follows: choose $r \leftarrow \mathbb{Z}_q$ and output $(g^r, h^r, z_i^r M_{b_i}, (c_i d_i^\alpha)^r)$, where α is as above. As in the proof of Theorem 3, we now modify the experiment. Given a tuple (g, h, u, v) which is either a DDH tuple or a random tuple, we define the output of $\mathsf{LR}_{PK,\vec{b}}(i, M_0, M_1)$ to be $(u, v, \tilde{w} = u^{x_{i,0}} v^{y_{i,0}} M_{b_i}, \tilde{e} = u^{x_{i,1}+x_{i,2}\alpha} v^{y_{i,1}+y_{i,2}\alpha})$, where $\alpha \stackrel{\text{def}}{=} H(u, v, \tilde{w})$. Note that if (g, h, u, v) is a DDH tuple, then this results in a perfect simulation of the original experiment. On the other hand, if (g, h, u, v) is a random tuple, then, under the DDH assumption, the success of any PPT adversary cannot differ by a non-negligible amount from its success in the original experiment. As in the proof of Theorem 3, note that, in running the experiment, we can answer all of A's key exposure queries. Thus, the proof handles an adaptive adversary whose key exposure requests may be made based on all information seen up to that point.

Assume now that (g, h, u, v) is a random tuple and $\log_g h \neq \log_u v$ (this occurs with all but negligible probability). We claim that, with all but negligible probability, the adversary's view in the modified experiment is independent of $\vec{b}$. The proof parallels [11, Lemma 2]. Say a ciphertext $\langle i, (u', v', w', e')\rangle$ is *invalid* if $\log_g u' \neq \log_h v'$. We have:

Claim: *If the decryption oracle outputs $\perp$ for all invalid ciphertexts during the adversary's attack, then the distribution of b_i (and hence $\vec{b}$) is independent of the adversary's view.*

The adversary knows at most $t-2$ values of $f_{x_0}(\cdot)$ and $f_{y_0}(\cdot)$ (at points other than i) and additionally knows the values $x^*_{1,0}$ and $y^*_{1,0}$ (the linear terms of these polynomials). Since $f_{x_0}(\cdot)$ and $f_{y_0}(\cdot)$ are random polynomials of degree t, the values $x_{i,0}, y_{i,0}$ $(= f_{x_0}(i), f_{y_0}(i))$ are uniformly distributed subject to:

$$\log_g z_i = x_{i,0} + y_{i,0} \log_g h. \tag{3}$$

Furthermore, when the decryption oracle decrypts valid ciphertexts $\langle i, (u', v', w', e')\rangle$, the adversary only obtains the linearly-dependent relations $r' \log_g z_i = r' x_{i,0} + r' y_{i,0} \log_g h$ (where $r' \stackrel{\text{def}}{=} log_g u'$). Similarly, decryptions of valid ciphertexts at other time periods do not further constrain $x_{i,0}, y_{i,0}$. Now consider the third component $u^{x_{i,0}} v^{y_{i,0}} M_{b_i}$ of the encryption oracle (the only one which depends on b_i). Since:

$$\log_u(u^{x_{i,0}} v^{y_{i,0}}) = x_{i,0} + y_{i,0} \log_u v, \tag{4}$$

and (3) and (4) are linearly independent, the conditional distribution of $u^{x_{i,0}} v^{y_{i,0}}$ (conditioned on b_i and the adversary's view) is uniform. Thus, the adversary's view is independent of b_i.

The following claim completes the proof of bidirectional security:

Claim: *With all but negligible probability, the decryption oracle will output $\perp$ for all invalid ciphertexts.*

Consider a ciphertext $\langle j, (u', v', w', e')\rangle$, where j represents a period during which a key exposure request was not made. We show that, with all but negligible probability, this ciphertext is rejected if it is invalid. There are two cases to consider: (1) $j = i$ (recall that i is the period during which the call to the LR oracle is made) and (2) $j \neq i$.

When $j = i$, the proof of the claim follows the proof of [11, Claim 2] exactly. The adversary knows at most $t-2$ values of $f_{x_1}(\cdot), f_{y_1}(\cdot), f_{x_2}(\cdot)$, and $f_{y_2}(\cdot)$ (at points other than i) and additionally knows the linear coefficients of these polynomials. Since these are all random polynomials of degree t, the values $(x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2})$ are uniformly distributed subject to:

$$\begin{aligned} \log_g c_i &= x_{i,1} + y_{i,1} \log_g h && (5) \\ \log_g d_i &= x_{i,2} + y_{i,2} \log_g h && (6) \\ \log_u \tilde{e} &= x_{i,1} + \alpha x_{i,2} + (\log_u v)\, y_{i,1} + (\log_u v)\, \alpha\, y_{i,2}, && (7) \end{aligned}$$

where (7) comes from the output of the encryption oracle. If the ciphertext $\langle i, (u', v', w', e')\rangle$ submitted by the adversary is invalid and $(u', v', w', e') \neq (u, v, \tilde{w}, \tilde{e})$, there are three possibilities:

Case 1. $(u', v', w') = (u, v, \tilde{w})$. In this case, $v' \neq \tilde{v}$ ensures that the decryption oracle will reject.

Case 2. $(u', v', w') \neq (u, v, \tilde{w})$ but $H(u', v', w') = H(u, v, \tilde{w})$. This violates the security of the universal one-way hash family and hence cannot occur with non-negligible probability. See [11].

Case 3. $H(u', v', w') \neq H(u, v, \tilde{w})$. The decryption oracle will reject unless:

$$\log_{u'} e' = x_{i,1} + \alpha' x_{i,2} + (\log_{u'} v')\, y_{i,1} + (\log_{u'} v')\, \alpha'\, y_{i,2}. \quad (8)$$

But (5)–(8) are all linearly independent, from which it follows that the decryption oracle rejects except with probability $1/q$. (As in [11], each rejection further constrains the values $(x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2})$; however, the k^{th} query will be rejected except with probability at most $1/(q - k + 1)$.)

When $j \neq i$, the values $(x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}, x_{j,1}, y_{j,1}, x_{j,2}, y_{j,2})$ are uniformly distributed subject only to (5)–(7) and:

$$\begin{aligned} \log_g c_j &= x_{j,1} + y_{j,1} \log_g h && (9) \\ \log_g d_j &= x_{j,2} + y_{j,2} \log_g h. && (10) \end{aligned}$$

Here, we make crucial use of the fact that the adversary has made at most $t - 2$ key exposure requests — had we allowed the adversary to learn $t - 1$ points on the polynomials, this (along with knowledge of the linear coefficients) would yield additional linear relations between, e.g., $x_{i,1}$ and $x_{j,1}$ and the proof of security would not go through.

If the ciphertext $\langle j, (u', v', w', e')\rangle$ submitted by the adversary is invalid, the decryption oracle will reject unless:

$$\log_{u'} e' = x_{j,1} + \alpha' x_{j,2} + (\log_{u'} v')\, y_{j,1} + (\log_{u'} v')\, \alpha'\, y_{j,2}. \quad (11)$$

Clearly, however, (5)–(7) and (9)–(11) are all linearly independent, from which it follows that the decryption oracle rejects except with probability $1/q$. This completes the proof of $(t - 2, N)$-bidirectional security.

The key to the proof above (informally) is that the adversary learns only $t - 1$ "pieces of information" about the polynomials $f_{x_1}(\cdot), f_{y_1}(\cdot), f_{x_2}(\cdot)$, and $f_{y_2}(\cdot)$ (i.e., their values at $t - 2$ points and their linear coefficients). Hence, before any calls to the decryption oracle have been made, the pair $(x_{i,1}, x_{j,1})$ (for example) is uniformly distributed. The proof of *strong* bidirectional security follows exactly the same arguments given above once we notice that SK^* gives only $t - 1$ "pieces of information" as well (i.e., the $t - 1$ leading coefficients). Details omitted. ■

We note that a trivial modification to the scheme achieves $(t - 1, N)$-bidirectional security with minimal added complexity: choose random elements $\{\tilde{x}_{1,n}, \tilde{y}_{1,n}\}_{0 \leq n \leq 2}$, then set $\hat{x}_{1,n} := x_{1,n} + \tilde{x}_{1,n}$ and $\hat{y}_{1,n} = y_{1,n} + \tilde{y}_{1,n}$ for $0 \leq n \leq 2$. Now, include $\{\tilde{x}_{1,n}, \tilde{y}_{1,n}\}_{0 \leq n \leq 2}$ with SK^* and store $\{\hat{x}_{1,n}, \hat{y}_{1,n}\}_{0 \leq n \leq 2}$ as part of SK_0 (and have these values be part of SK_i at all time periods). Key updates are done in the obvious way. Note that SK^* only stores $t - 1$ "pieces of information" about the random, degree-t polynomials; furthermore, $t - 1$ key exposures only reveal $t - 1$ "pieces of information" as well. Thus, a proof of security follows the proof of the above theorem.

References

[1] M. Abdalla and L. Reyzin. A New Forward-Secure Digital Signature Scheme. Asiacrypt '00.

[2] M. Abe and M. Kanda. A Key Escrow Scheme with Time-Limited Monitoring for One-Way Communication. ACISP ’00.

[3] R. Anderson. Invited lecture. ACM CCCS ’97.

[4] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. FOCS ’97.

[5] M. Bellare and S.K. Miner. A Forward-Secure Digital Signature Scheme. Crypto ’99.

[6] G. Blakley and C. Meadows. Security of Ramp Schemes. Crypto ’84.

[7] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. Crypto ’01.

[8] V. Boyko. On the Security Properties of the OAEP as an All-or-Nothing Transform. Crypto ’99.

[9] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-Or-Nothing-Transforms. Eurocrypt ’00.

[10] B. Chor, A. Fiat, and M. Naor. Tracing Traitors. Crypto ’94.

[11] R. Cramer and V. Shoup. A Practical Public-Key Cryptosystem Provably Secure against Adaptive Chosen-Ciphertext Attacks. Crypto ’98.

[12] A. Dyachkov and V. Rykov. A Survey of Superimposed Code Theory. In *Problems of Control and Information Theory*, vol. 12, no. 4, 1983.

[13] T. El Gamal. A Public-Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm. *IEEE Transactions of Information Theory*, 31(4): 469–472, 1985.

[14] P. Erdos, P. Frankl, and Z. Furedi. Families of Finite Sets in which no Set is Covered by the Union of r Others. In *Israel J. Math.*, 51(1-2): 79–89, 1985.

[15] M. Franklin, M. Yung. Communication Complexity of Secure Computation. STOC ’92.

[16] E. Gafni, J. Staddon, and Y. L. Yin. Efficient Methods for Integrating Traceability and Broadcast Encryption. Crypto ’99.

[17] M. Girault. Relaxing Tamper-Resistance Requirements for Smart Cards Using (Auto)-Proxy Signatures. CARDIS ’98.

[18] O. Goldreich, B. Pfitzmann, and R.L. Rivest. Self-Delegation with Controlled Propagation — or — What if You Lose Your Laptop? Crypto ’98.

[19] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Computing 17(2): 281–308 (1988).

[20] P. Indyk. Personal communication.

[21] G. Itkis and L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. Crypto ’01.

[22] S. Jarecki and A. Lysyanskaya. Concurrent and Erasure-Free Models in Adaptively-Secure Threshold Cryptography. Eurocrypt ’00.

[23] H. Krawczyk. Secret Sharing Made Short. Crypto ’93.

[24] H. Krawczyk. Simple Forward-Secure Signatures From any Signature Scheme. ACM CCCS ’00.

[25] R. Kumar, S. Rajagopalan, and A. Sahai. Coding Constructions for Blacklisting Problems without Computational Assumptions. Crypto ’99.

[26] C.-F. Lu and S.W. Shieh. Secure Key-Evolving Protocols for Discrete Logarithm Schemes. RSA 2002, to appear.

[27] T. Malkin, D. Micciancio, and S. Miner. Composition and Efficiency Tradeoffs for Forward-Secure Digital Signatures. Available at `http://eprint.iacr.org`.

[28] T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. Crypto '91.

[29] R. Rivest. All-or-Nothing Encryption and the Package Transform. FSE '97.

[30] A. Shamir. How to share a secret. *Comm. ACM*, 22(11):612–613, 1979.

[31] W.-G. Tzeng and Z.-J. Tzeng. Robust Key-Evolving Public-Key Encryption Schemes. Available at `http://eprint.iacr.org`.

[32] —. Bidirectionally-Secure Signature Schemes. Manuscript, 2001.

A Bidirectional Security of Signature Schemes

In this section, we provide a formal model and definition for key-updating signature schemes, and bidirectional security of such schemes. Not surprisingly, the treatment is very similar to that of Section 2.

Definition 6 A key-updating signature scheme is a 4-tuple of poly-time algorithms $(\mathcal{G}, \mathcal{U}^*, \mathcal{U}, \mathsf{Sign}, \mathsf{Vrfy})$ such that:

- $\mathcal{G}$, the *key generation algorithm*, is a probabilistic algorithm which takes as input a security parameter 1^k and the total number of time periods N. It returns a public key PK, a *master key* SK^*, and an initial key SK_0.
- $\mathcal{U}^*$, the *device key-update algorithm*, is a deterministic algorithm which takes as input an index i for a time period (throughout, we assume $1 \le i \le N$) and the master key SK^*. It returns (partial) secret key SK'_i for time period i.
- $\mathcal{U}$, the *user key-update algorithm*, is a deterministic algorithm which takes as input an index i, secret key SK_{i-1}, and (partial) secret key SK'_i. It returns secret key SK_i for time period i.
- Sign, the *signing algorithm*, takes as input an index i of a time period, a message M, and a secret key SK_i corresponding to period i. $\mathsf{Sign}_{SK_i}(i, M)$ returns a signature $\langle i, s \rangle$ consisting of the time period i and a tag s. This algorithm may be probabilistic.
- Vrfy, the *verification algorithm*, is a deterministic algorithm which takes as input the public key PK, a message M, and a pair $\langle i, s \rangle$. $\mathsf{Vrfy}_{PK}(M, \langle i, s \rangle)$ returns a bit b, where $b = 1$ means the signature is accepted.

If $\mathsf{Vrfy}_{PK}(M, \langle i, s \rangle) = 1$, we say that $\langle i, s \rangle$ is a *valid* signature of M for period i. We require that all signatures output by $\mathsf{Sign}_{SK_i}(i, M)$ are accepted as valid by Vrfy. ■

The security of key-updating signature scheme is again defined very similarly to Definitions 2-4, except we consider existential unforgeability against adaptive chosen message attack [19]. We briefly sketch these definitions. In all of them, the attacker has access to the signing oracle $\mathsf{Sign}(\cdot, \cdot)$, which on input (i, M) returns the signature of M at period i. For the analog of Definition 2 ((t, N)-bidirectional security), the adversary also gets the key exposure oracle $\mathsf{Exp}(\cdot)$ which it can call t times (and which on input i, returns the secret key SK_i). The goal of the adversary is to output a valid signature pair $\langle i, s \rangle$ of a "new" message M, i.e. (1) $\mathsf{Vrfy}_{PK}(M, \langle i, s \rangle) = 1$; (2) (i, M) was never queried to the signing oracle; and (3) the adversary did not query the key exposure oracle at period i. The key-updating signature scheme is (t, N)-bidirectionally-secure if any PPT adversary has a negligible probability of success. In the analog of Definition 4 (strong bidirectional security), instead of the key exposure oracle the adversary gets the master key SK^* (and condition (3) above is no longer relevant). Finally, Definition 3 (secure key updates) does not change.

In a companion paper [32], we present a number of implementations of this new notion.